2018 빅콘테스트
Analysis 분야 챔피언리그

- 게임 유저 이탈 예측 모형 -

2018. 09. 14

팀명: 투빅스

팀원: 이경택, 최희정, 김민정, 김소희, 조양규

NCSOFT®

2018 빅콘테스트
2018 BIG CONTEST

AGENDA

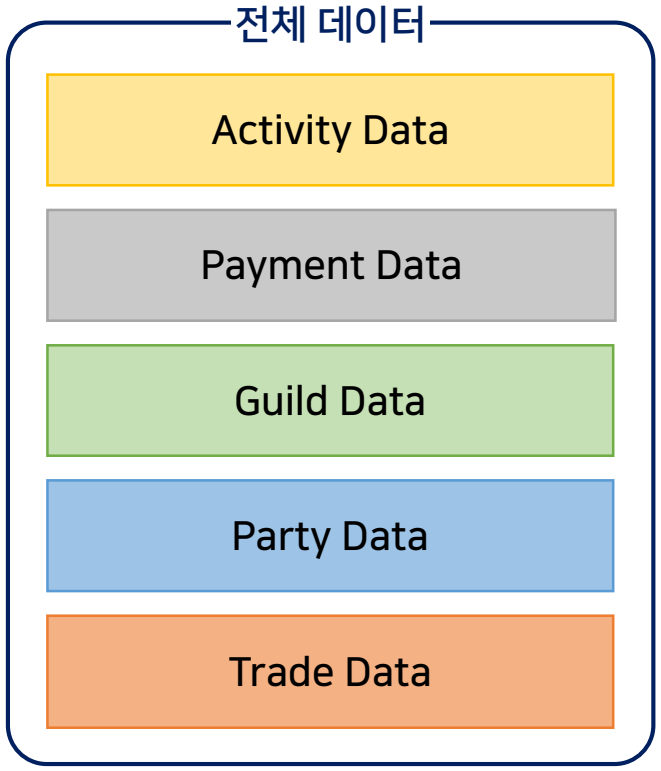
1. 데이터 전처리
2. 1차 모델링
3. 2차 모델링
4. 결과 및 해석

AGENDA

1. 데이터 전처리
2. 1차 모델링
3. 2차 모델링
4. 결과 및 해석

문제 정의 및 데이터 소개

- 유저들의 게임 활동 정보를 이용하여 유저가 향후 게임 서비스에서 이탈하는 시점을 예측하는 모형 개발



분석에 앞서 분석의 성능을 높이기 위한
파생변수 생성 및 데이터 정제 실시

Activity Data

- 파생 변수 생성

  - 1) 승률
    - 결투와 전장에 대한 참여 횟수 대비 승리 횟수를 나타내는 변수 생성

  - 2) 성과
    - NPC 사냥과 퀘스트를 통해 얻은 총 경험치를 나타내는 변수 생성

  - 3) Clear 비율
    - 인던, 레이드, 평야에 대한 입장 횟수 대비 clear 횟수를 나타내는 변수 생성
    - 솔로, 라이트, 숙련별로 총 8개의 clear 비율 관련 파생 변수 생성

Activity Data

▪ 파생 변수 생성

파생 변수 종류	파생 변수 의미	파생 변수 생성 공식
승률	결투 승률($duel\_per$), 전장 승률($partybattle\_per$)	$duel\_per = \frac{duel\_win}{duel\_cnt}$, $partybattle\_per = \frac{partybattle\_win}{partybattle\_cnt}$
성과	총 경험치(get_exp)	$get\_exp = npc\_exp + npc\_hongmun + quest\_exp + quest\_hongmun + item\_hongmun$
Clear 비율	솔로 인던 완료 비율($prob\_clear\_inzone\_solo$)	$prob\_clear\_inzone\_solo = \frac{cnt\_clear\_inzone\_solo}{cnt\_enter\_inzone\_solo}$
	⋮	⋮
	밤의 바람 평야 완료 비율($prob\_clear\_bam$)	$prob\_clear\_bam = \frac{cnt\_clear\_bam}{cnt\_enter\_bam}$

정제 Data

	Original Activity Data				승률		성과	Clear 비율(인던, 레이드, 평야)		
	wk	acc_id	cnt_dt	…	making_cnt	duel_per	partybattle_per	get_exp	prob_clear_inzone_solo	…	prob_clear_bam
	7	3dc6f2875d…	4		-0.36554	1.1426	0.9546	11.3170	0.9927		0.6954
	8	3dc6f2875d…	5		-0.36554	1.1426	0.9546	15.0973	0.9927		0.6954
	3	b8856358ff…	2		-0.36554	1.1426	0.9546	-1.5618	0.9927		0.6954

Activity Data

- 시계열 데이터 정제
  - 각 acc_id에 대한 original data의 week 1~8주 데이터를 행을 기준으로 이어 붙여 시계열 데이터의 특징을 반영함
  - 게임에 참여하지 않은 week의 경우 결측치를 대체하기 위해 표준화된 데이터의 특징을 고려해 해당 열의 최솟값으로 대체함

Original Data

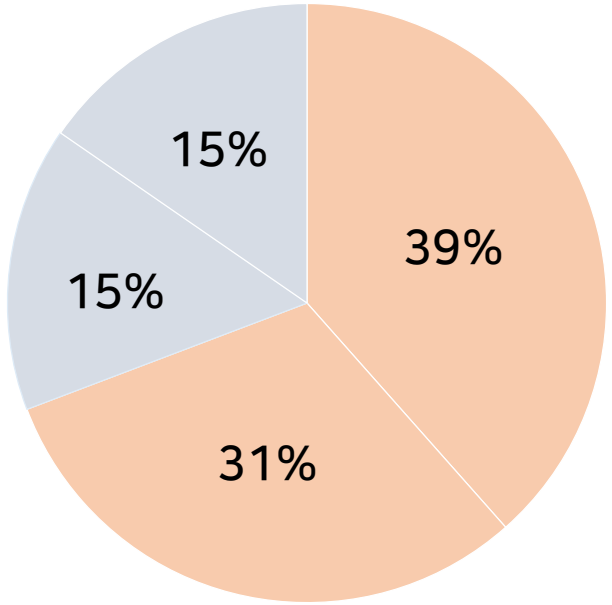
wk	acc_id	cnt_dt	play_time	npc_exp	···	prob_clear_bam
1	23becd...	4				
7	23becd...	5				
8	23becd...	2				
1	172b44...	2				

정제 Data

	week1			week2				week8		
acc_id	cnt_dt	···	prob_clear_bam	cnt_dt	···	prob_clear_bam	···	cnt_dt	···	prob_clear_bam
23becd...	4			min			···	2		
172b44...	2			1				3		

Activity Data & Payment Data

- Idea



52명 대상 게임 이탈 원인 설문조사 결과

- 투자한 비용대비 능력치가 낮아서 V
- 투자한 시간대비 능력치가 낮아서 V
- 친구들이 그만두어서
- 게임에 버그가 많아서

- Activity Data의 play_time 변수와 Payment Data의 payment_amount 변수와 관련된 응답이 상위권을 차지함

Activity Data & Payment Data

설문조사 결과를 바탕으로 파생 데이터 생성

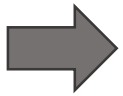
- 1) 시간 대비 Data
  - play_time 투자 대비 성과(승리/경험치…)가 좋지 않을 경우 이탈하는 특성을 반영한 데이터
- 2) 비용 대비 Data
  - payment_amount 투자 대비 성과가 좋지 않을 경우 이탈하는 특성을 반영한 데이터
- 3) 시간 가중 Data
  - play_time을 많이 투자할수록 이탈하지 않는 특성을 반영한 데이터
- 4) 비용 가중 Data
  - payment_amount를 많이 투자할수록 이탈하지 않는 특성을 반영한 데이터
- 5) 시간 및 비용 대비 Data
  - play_time과 payment_amount 투자 대비 성과가 좋지 않을 경우 이탈하는 특성을 반영한 데이터
- 6) 시간 및 비용 가중 Data
  - play_time과 payment_amount를 많이 투자할수록 이탈하지 않는 특성을 반영한 데이터

Activity Data & Payment Data

- Play_time과 payment_amount 변수 보정

< original payment data >

payment_week	acc_id	payment_amount
1	3dc6f2875d...	-0.1499
3	b7b090a585...	5.123541
6	316f50d335...	16.72511



< preprocessed payment data >

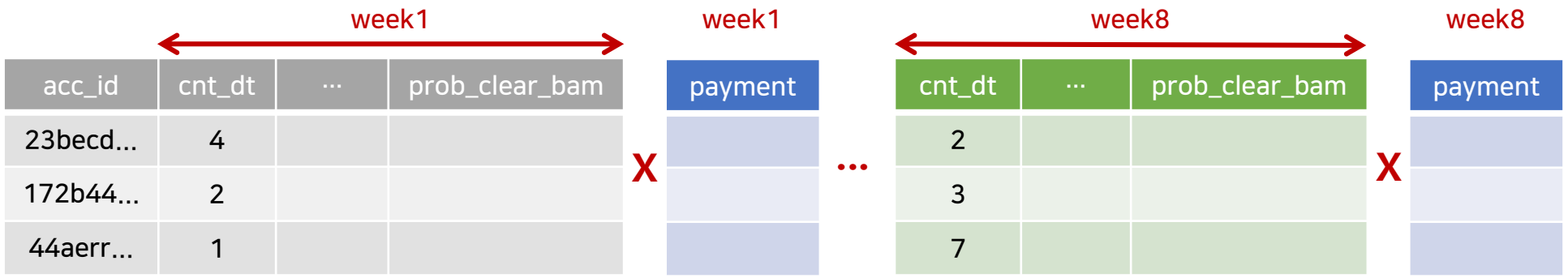
payment_week	acc_id	payment_amount
1	3dc6f2875d...	1
3	b7b090a585...	6.273441
6	316f50d335...	17.87501

- 기존 데이터는 표준화된 데이터로 play_time과 payment_amount 변수가 음수 값을 가짐
- 따라서, play_time과 payment_amount 변수를 이용하여 연산하는 과정에서 문제가 발생함
  - ex) payment _amount(-)와 activity data(-)가 곱해질 경우 값이 (+)가 되어 상반된 의미가 됨
- 위의 문제점을 해결하기 위해 곱셈과 나눗셈 연산의 특성을 고려해 변수 play_time과 payment_amount의 최솟값이 1이 되도록 각 변수의 값을 shift 시킴

Activity Data & Payment Data

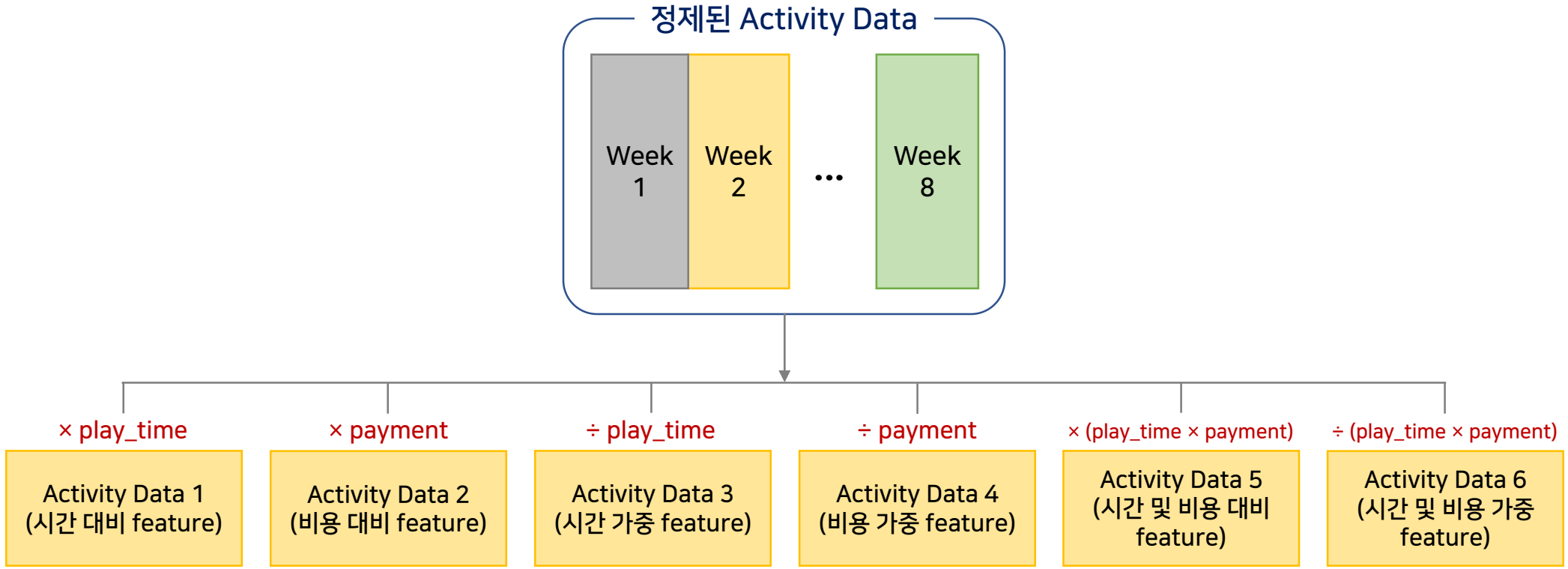
- 6개의 파생 데이터 생성

ex) 비용 대비 Data 생성



- 정제한 activity data에 각 주에 해당하는 play_time과 payment_amount 변수 값을 곱하거나 나누는 연산을 통해 각각 다른 의미를 나타내는 6개의 파생 데이터를 생성함

Activity Data & Payment Data



- Activity Data로부터 각각 다른 의미를 지닌 6개의 파생 데이터를 생성함

Guild Data

- Guild Data의 feature 생성

  - 1) one_guild
    - acc_id를 기준으로 해당 acc_id가 guild 인원수가 1명인 1인 guild에 참여했는가의 여부를 나타내는 변수 생성(1인 guild 참여시 1 / 미참여시 0)

  - 2) num_guild_member
    - acc_id를 기준으로 해당 acc_id가 참여한 guild의 평균 member 수 변수 생성

  - 3) num_guild
    - acc_id를 기준으로 해당 acc_id가 참여한 guild의 총 개수 변수 생성

Guild Data

- Acc_id 기준 Guild Data 정제
  - 유저가 guild에 참여하지 않아 발생한 결측치는 0으로 대체함

Original Data

guild_id	guild_member_acc_id
0015cb37ae...	9eee6b10b8...
	a767abcfbd...
	b178c21c7e...
000b682e9e...	000b682e9e...

정제 Data

acc_id	one_guild	num_guild_member	num_guild
B8fbf3f6a7...	0	9	1
Ed500c495...	0	15	3
Acc6afa23a...	1	1	1

Party Data

- Party Data의 feature 생성

  - 1) party_time
    - acc_id를 기준으로 해당 acc_id가 party에 참여한 평균 시간 변수 생성 (단위: 분)

  - 2) num_ party_ member
    - acc_id를 기준으로 해당 acc_id가 참여한 party의 평균 member 수 변수 생성

  - 3) num_party
    - acc_id를 기준으로 해당 acc_id가 참여한 party의 총 개수 변수 생성

Party Data

- Acc_id 기준 Party Data 정제
  - 유저가 party에 참여하지 않아 발생한 결측치는 0으로 대체함

Original Data

start_week	start_day	start_time	end_week	end_day	end_time	members_acc_id
1	1	09:14:58.558	1	1	09:41:30.200	11fc85879e...
3	3	11:05:05.176	3	3	13:07:42.515	7176c15162...
3	6	02:18:43.172	3	6	02:28:58.177	8092e194a7...

정제 Data

acc_id	party_time	num_party_member	num_party
B8fbf3f6a7...	3.435644	5.737624	202
Ed500c495...	13.03488	3.534884	172
Acc6afa23a...	2.452555	4.927007	137

Trade Data

- Trade Data의 feature 생성

  - 1) item_type (accessory / costume / gem / grocery / money / weapon)
    - acc_id를 기준으로 해당 acc_id가 판매자로 참여한 trade의 item별 총 amount 변수 생성

  - 2) num_ trade
    - acc_id를 기준으로 해당 acc_id가 판매자로 참여한 trade의 총 횟수 변수 생성

Trade Data

- Acc_id 기준 Trade Data 정제
  - 유저가 trade에 참여하지 않아 발생한 결측치는 0으로 대체함

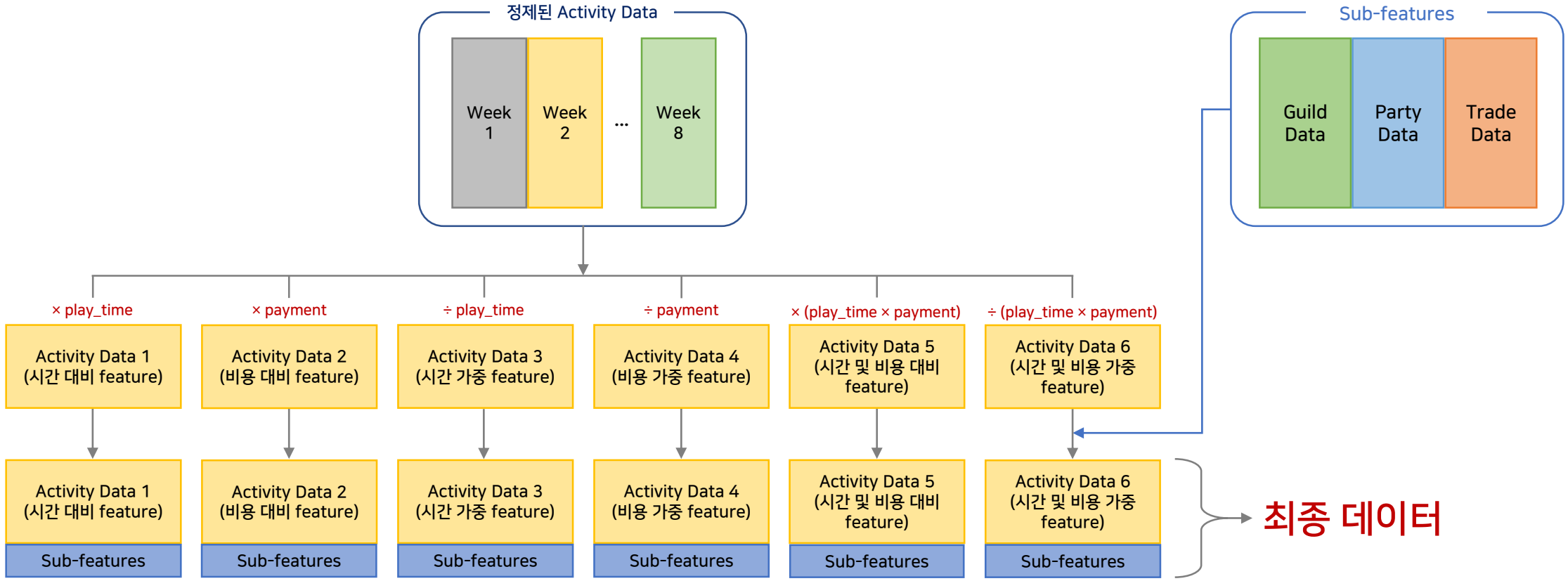
Original Data

trade_week	trade_day	trade_time	source_acc_id	target_acc_id	item_type	item_amount
8	2	23:32:06	3dc6f2875d...	96995041e6...	grocery	-0.05634
4	6	13:59:30	fe5f27df6cf...	06fb9a2059...	accessory	-0.05635
7	6	23:12:15	02ab144aba...	d31b2c8d10...	costume	-0.05635
1	7	23:12:13	e1a0af6b1fb...	f860c1f4536...	gem	-0.05635

정제 Data

acc_id	accessory	gem	grocery	money	weapon	costume	num_trade
B8fbf3f6a7...	-0.6761	0	0	-0.5718	0	0	24
Ed500c495...	-0.1127	0	0	-0.1061	0	0	4
Acc6afa23a...	-3.7188	0	0	-3.6417	0	0	132

최종 데이터



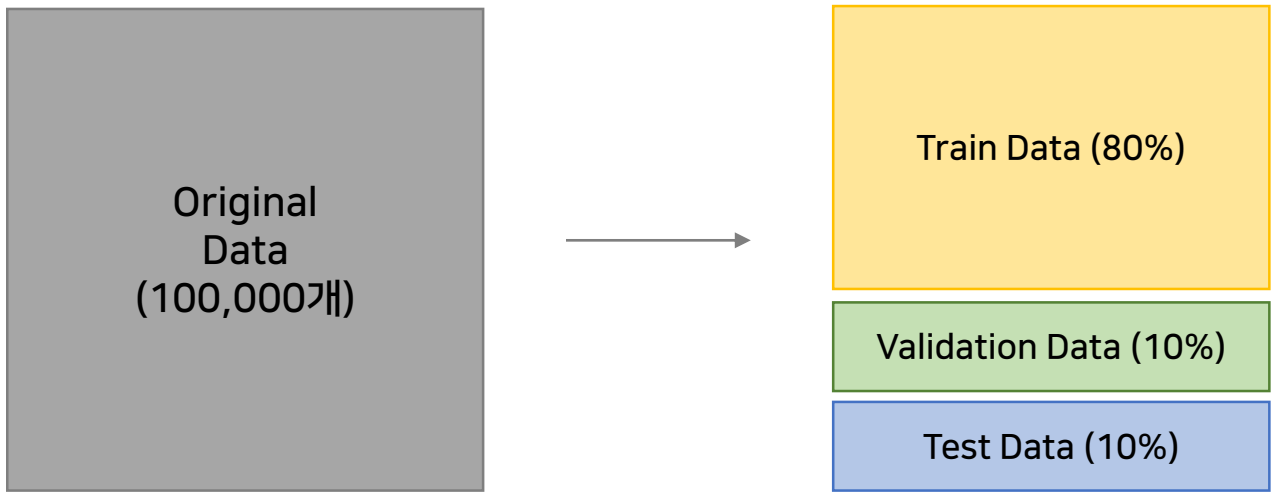
- Activity data와 Payment data를 통해 생성한 6가지 파생 데이터에 Sub-features로 정의한 정제된 Guild data, Party data, Trade data를 concatenate해서 최종 데이터를 생성함

AGENDA

1. 데이터 전처리
2. 1차 모델링
3. 2차 모델링
4. 결과 및 해석

Data Split

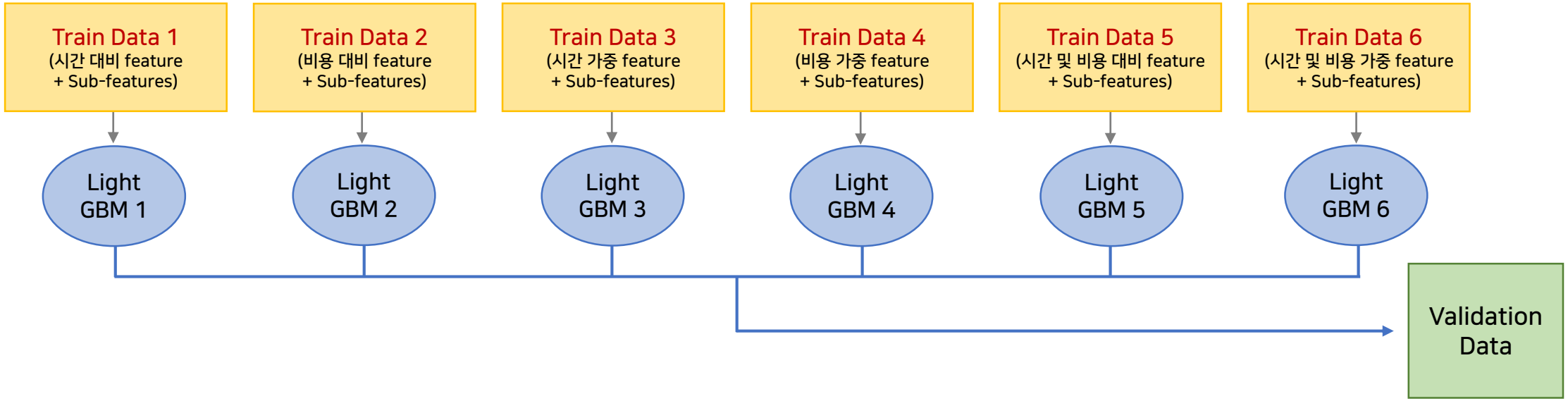
- Data Split



- 전체 데이터를 train : validation : test 데이터의 비율이 8:1:1이 되도록 분할함
- Train data로 학습한 모델을 validation data에 test하며 최적의 parameter를 탐색함
- 구축된 모델을 test data에 적용해 최종 모델 평가를 진행함

모델링 과정

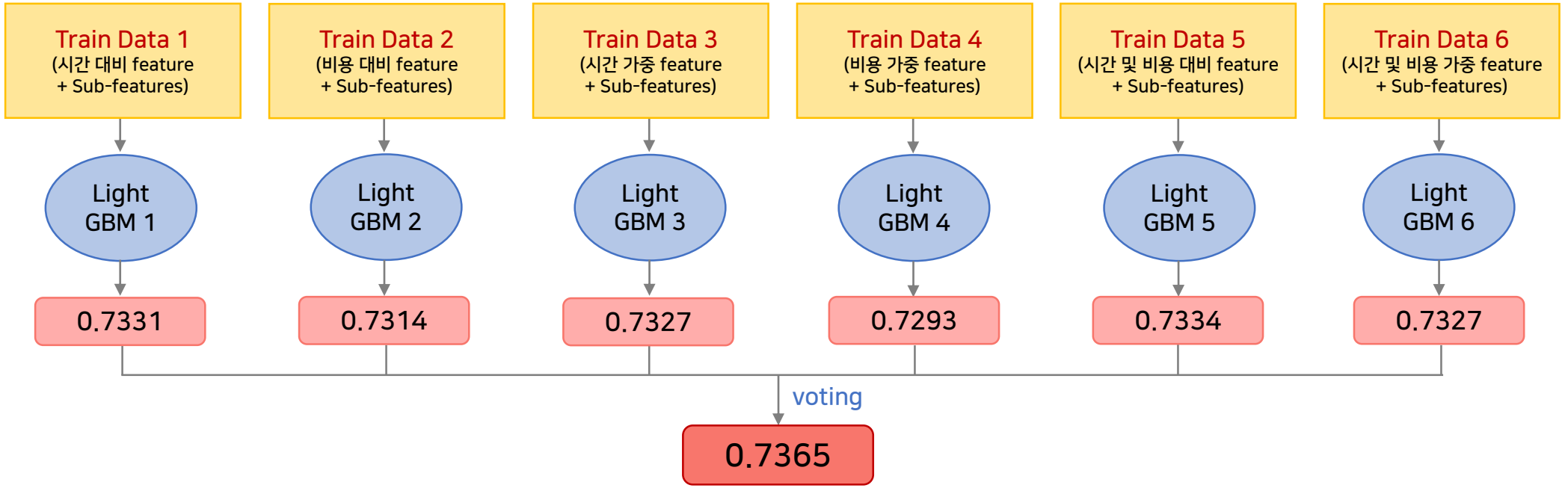
- 앙상블 모델 학습



- Step 1) i번째 train data를 이용해 i번째 Light GBM 모델을 학습함
- Step 2) step 1을 6회 반복하여 총 6개의 Light GBM 단일 모델을 구축함
- Step 3) 6개의 Light GBM 모델의 voting을 통해 validation data의 class를 예측함

모델링 결과

- 앙상블 모델 결과



- 각각의 단일 모델은 0.7310~0.7334 사이의 F1-measure 결과 값을 가짐
- 단일 모델의 앙상블을 통해 구축된 최종 모델은 단일 모델보다 향상된 0.7365의 F1-measure 결과 값을 가짐

AGENDA

1. 데이터 전처리
2. 1차 모델링
3. 2차 모델링
4. 결과 및 해석

분석 도메인 특징

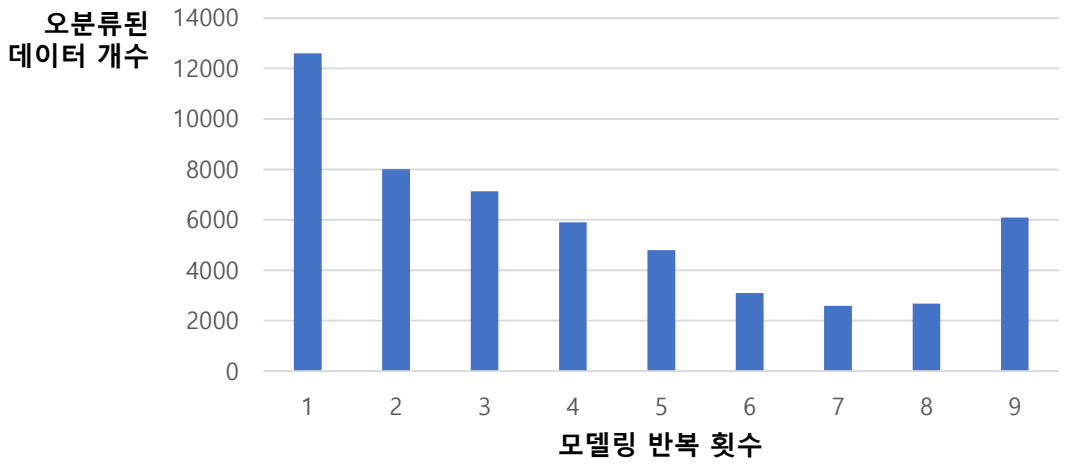
- 1. month vs 2month 예측 문제

Pred \ True	week	month	2month	retained	sum
week	2091	153	91	177	2512
month	105	1591	582	250	2528
2month	74	471	1746	201	2492
retained	250	88	243	1887	2468
sum	2520	2303	2662	2515	10000

- Validation data에 대한 confusion matrix 확인 결과, 모델이 month와 2month 분류에 어려움이 있는 것을 확인할 수 있음
- month class로 예측된 데이터의 개수가 상대적으로 적은 것을 통해 모델이 month class에 대해 underfitting 됐다고 볼 수 있음

분석 도메인 특징

- 2. 오분류된 데이터의 반복적인 오분류 현상



- 오분류된 train data의 샘플링 가중치를 높여 모델을 재학습하고 train data를 예측함
- 위의 과정을 반복 시행한 결과, 오분류된 데이터는 가중치를 주어 샘플링해도 지속적으로 오분류됨
- 즉, train data의 예측 정확도가 85% 이상까지 도달하지 못하므로 train data 내에도 15% 이상이 맞추기 어려운 데이터라고 볼 수 있음
- 더불어 test data의 정확도는 약 73%로 train data의 정확도와 큰 차이가 존재 하지 않음

모델 보완 전략

모델 보완 전략

1. month vs 2month 예측 문제

- 1) 첫번째 전략
  - month와 2month 데이터만을 이용해 두 class에 대한 Binary 분류 모델을 구축하는 전략
- 2) 두번째 전략
  - month와 2month 데이터에 가중치를 주는 샘플링을 통해 해당 class의 예측 확률을 높이는 전략

2. 오분류된 데이터의 반복적인 오분류 현상

- 3) 세번째 전략
  - confidence가 매우 높은 test data의 predicted label 값을 true label 값으로 가정하고 train data에 추가해 데이터를 증가시키는 전략

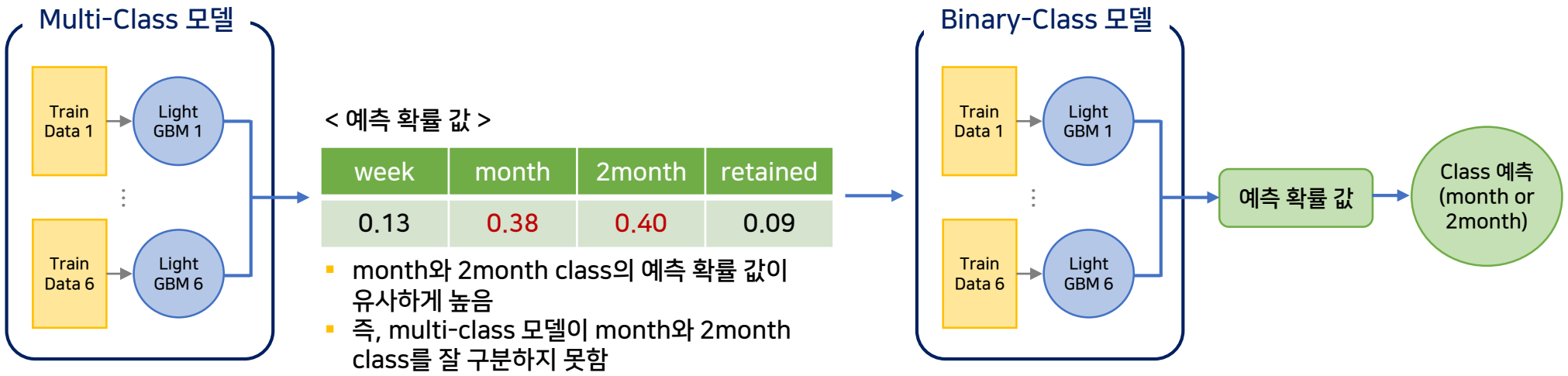
1. month vs 2month 예측 문제 – 첫번째 전략

- 1) Binary 분류 모델 구축

  - month와 2month class에 해당하는 데이터 중 20%를 샘플링함
  - 기존 모델링과 동일한 방식으로 데이터를 분할함

1. month vs 2month 예측 문제 – 첫번째 전략

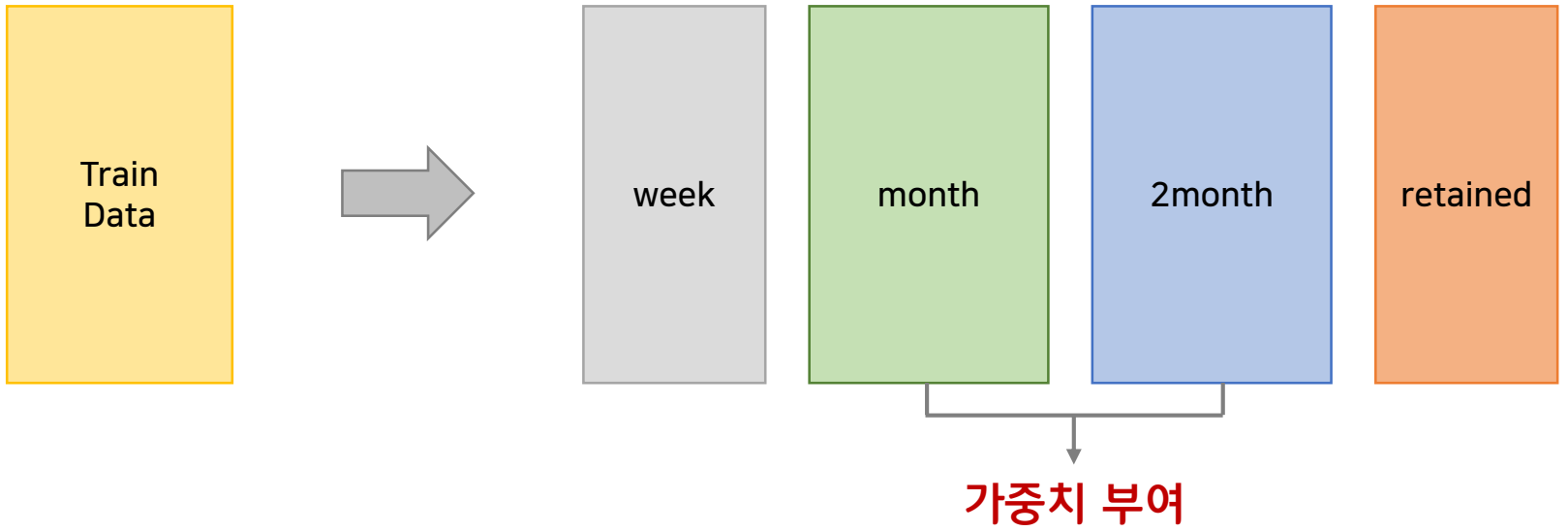
1) Binary 분류 모델 구축



- 기존 모델의 예측 결과가 month 또는 2month class일 때, 두 class에 대한 예측 확률 값이 비슷하면 사전에 학습한 Binary 분류 모델을 기반으로 최종 class를 예측함
- 모델이 분류하기 어려운 class에 대해 상대적으로 성능이 좋은 Binary 분류 모델을 구축해 해당 class에 대한 예측력을 높임

1. month vs 2month 예측 문제 – 두번째 전략

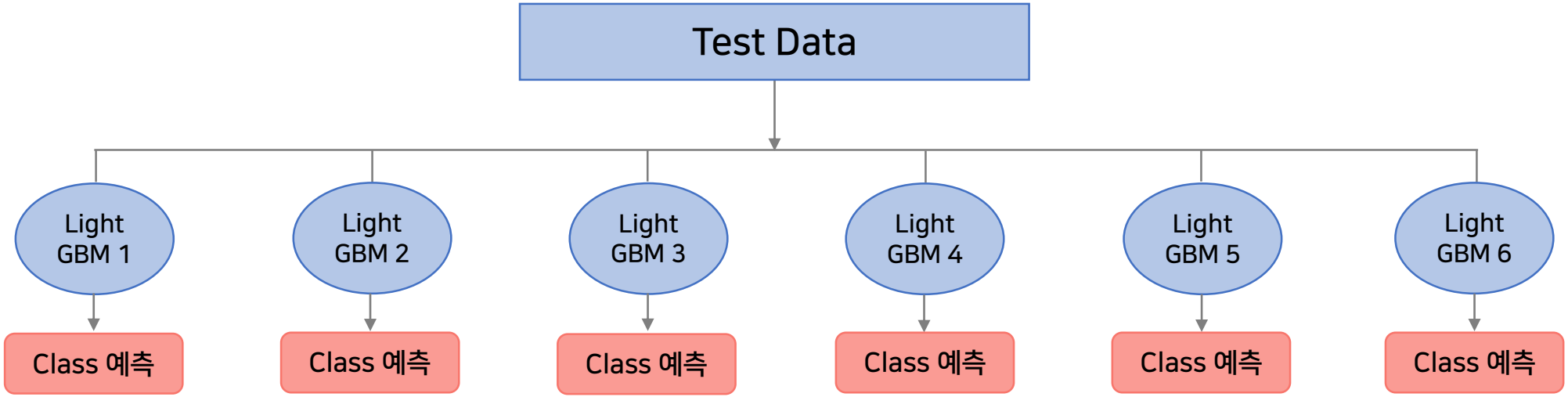
- 2) month & 2month에 가중치를 주는 샘플링



- 기존 모델의 예측 결과 분석 시, month와 2month class가 다른 class보다 예측 확률이 낮은 것을 확인할 수 있음
- 해당 문제점을 해결하기 위해 month와 2month class에 가중치를 주는 샘플링을 통해 train data를 구축하고 이를 기반으로 모델을 학습함

2. 오분류된 데이터의 반복적인 오분류 현상 – 세번째 전략

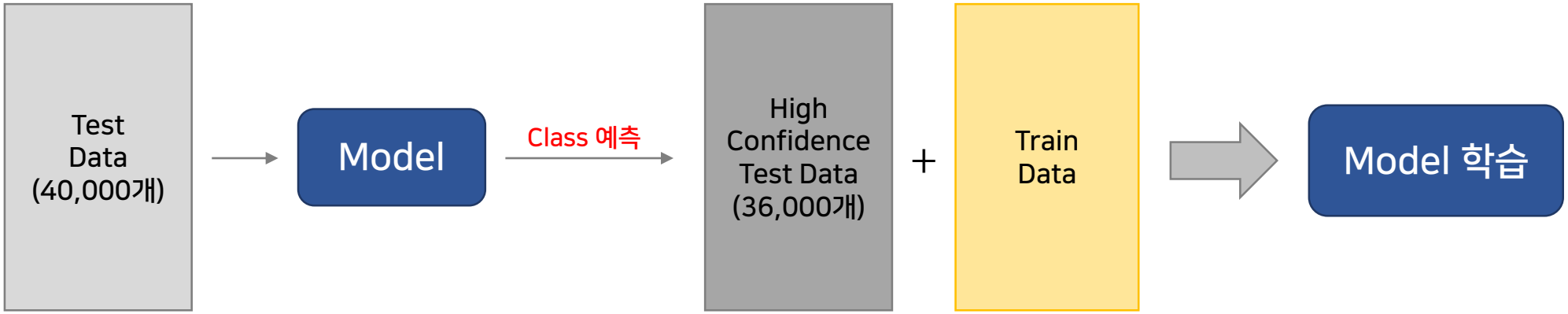
- 3) Confidence가 높은 test data를 train data에 추가



- Test data의 정확도는 약 73%로 train data의 정확도와 큰 차이가 존재 하지 않음
- 단일 모델의 F1-measure 결과 값은 0.7310~0.7334 사이로 높은 결과를 보임
- 따라서, 단일 모델 6개의 class 예측 값이 모두 일치하는 confidence가 매우 높은 test data는 predicted class 값을 true class 값으로 간주함 (약 80% 정확도 추정)

2. 오분류된 데이터의 반복적인 오분류 현상 – 세번째 전략

- 3) Confidence가 높은 test data를 train data에 추가



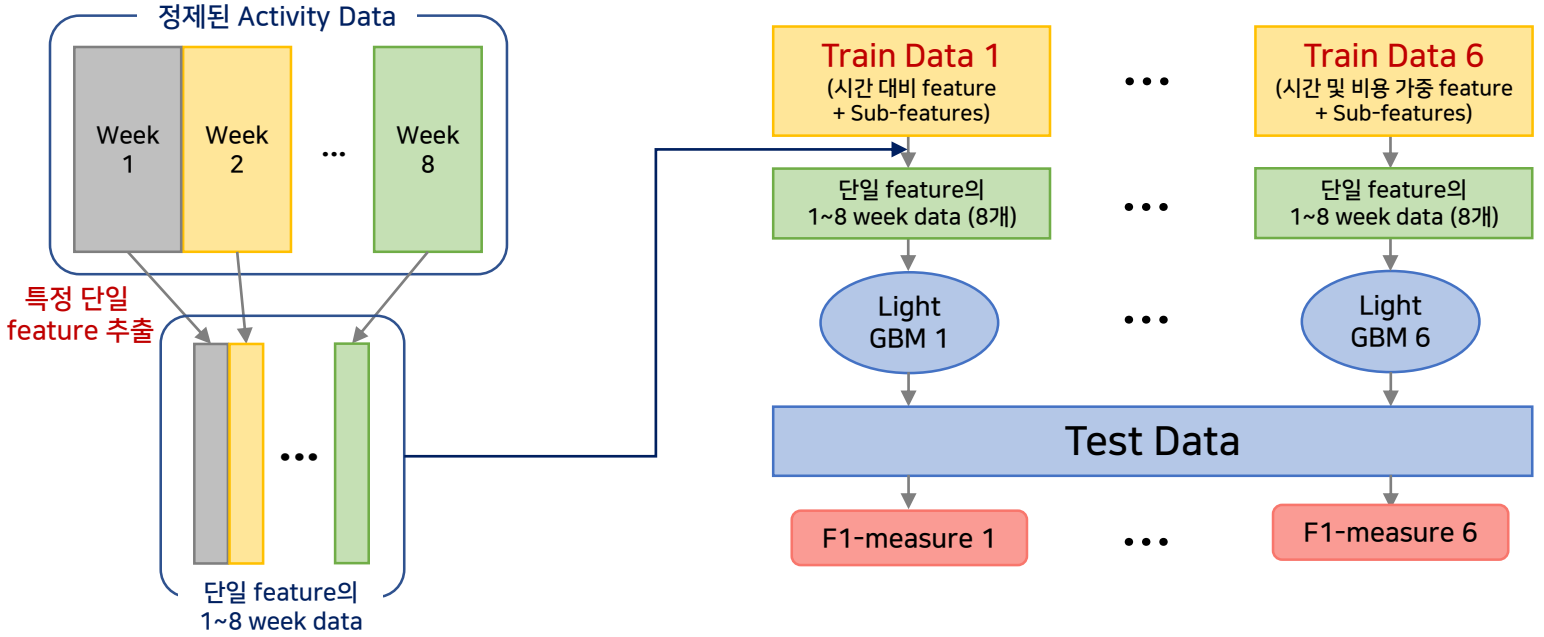
- 6개 단일 모델의 class 예측 값이 모두 동일한 confidence가 높은 test data의 predicted class를 true class로 가정함
- Train data에 위의 데이터를 추가한 데이터를 기반으로 기존 모델을 재학습함
- 데이터 개수의 증가를 통해 모델의 성능을 향상시킴 (0.2~0.3% 성능 향상)

AGENDA

1. 데이터 전처리
2. 1차 모델링
3. 2차 모델링
4. 결과 및 해석

결과 및 해석 1

- 단일 feature 중요도 실험



- 41개의 단일 feature 중 중요한 feature를 추출하기 위해 실험을 진행함
- Step 1) 각 train data에서 단일 feature의 1~8 week data를 추출함
- Step 2) Step 1에서 추출된 각 데이터로 6개의 모델을 각각 학습함
- Step 3) Step 2에서 학습한 모델을 test data에 적용해 각 모델의 F1-measure를 측정함

결과 및 해석 1

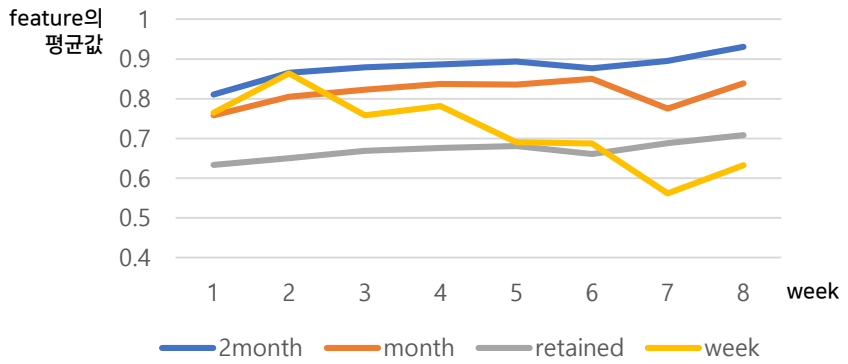
- 단일 feature 중요도 실험

data 종류 / 단일 feature	Acc of 시간 대비 feature	Acc of 비용 대비 feature	Acc of 시간 가중 feature	Acc of 비용 가중 feature	Acc of 시간 및 비용 대비 feature	Acc of 시간 및 비용 가중 feature
cnt_enter_raid	0.6865	0.4232	0.6704	0.4105	0.6799	0.6821
party_battle_win	0.6842	0.4387	0.6660	0.4172	0.6779	0.6783
duel_win	0.6787	0.4238	0.6685	0.4352	0.6754	0.6776
cnt_clear_raid_light	0.6898	0.4334	0.6688	0.4217	0.6790	0.6876
⋮	⋮	⋮	⋮	⋮	⋮	⋮
cnt_dt	0.5945	0.5859	0.5855	0.5903	0.5817	0.6013

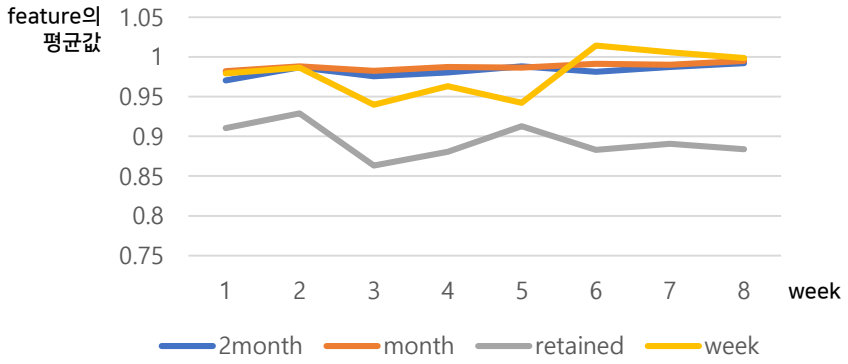
- 각 단일 feature들은 0.6~0.69 정도의 준수한 성능을 보임
- 특정 단일 변수가 매우 높은 성능을 보이지는 않았으나, 비용 대비 및 비용 가중 관련하여 전투 관련 feature(전투, 전장, 레이드)가 다른 feature 대비 매우 낮은 성능을 보임

결과 및 해석 1

- 단일 feature 중요도 실험



< 시간 대비 라이트 레이드 완료 확률 >

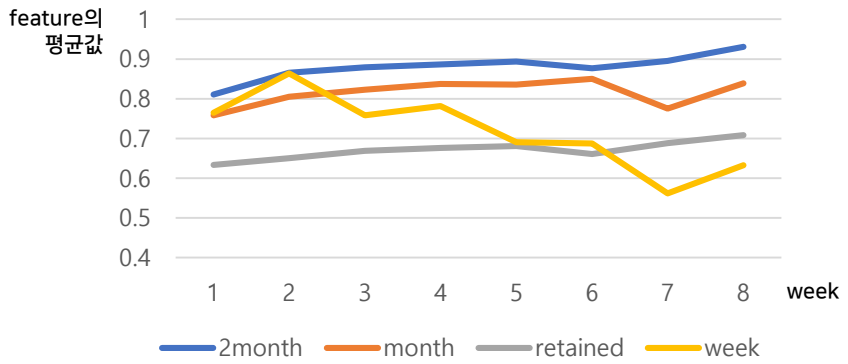


< 비용 대비 라이트 레이드 완료 확률 >

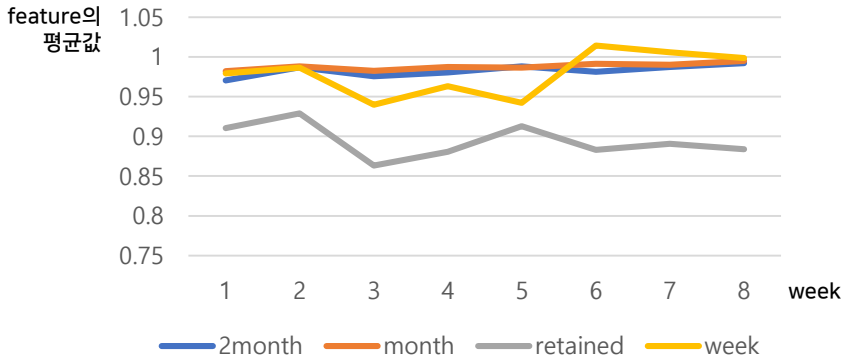
- 이전 단일 feature 중요도 실험에서 상위권에 해당하는 전투 관련 feature 중 라이트 레이드 완료 확률 feature에 대해 추가적으로 분석을 진행함
- 각 week별 시간 및 비용 대비 라이트 레이드 완료 확률의 평균값을 그래프로 나타냄
- 이탈(week, month, 2month)/잔존(retained) 두개의 큰 class를 기준으로 시간 및 비용 대비 라이트 레이드 완료 확률을 비교한 결과, 이탈과 잔존에서 해당 feature의 추세가 다르게 나타나는 것을 확인할 수 있음

결과 및 해석 1

- 단일 feature 중요도 실험



< 시간 대비 라이트 레이드 완료 확률 >

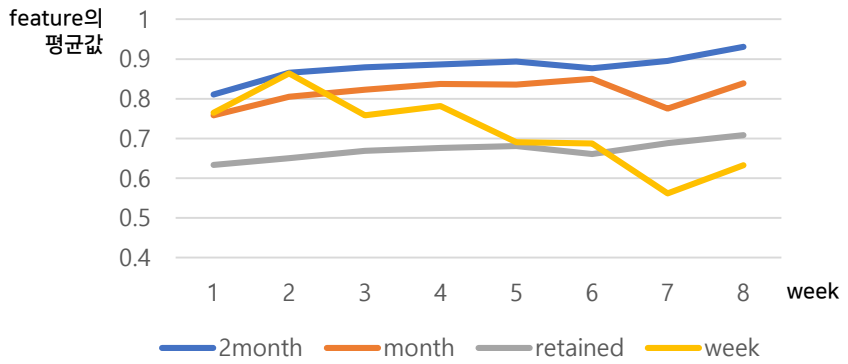


< 비용 대비 라이트 레이드 완료 확률 >

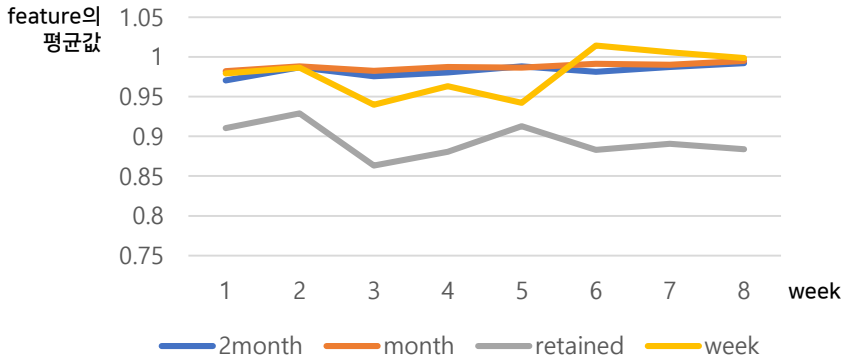
- 비용 대비 라이트 레이드 완료 확률의 경우, 이탈 class에 대한 해당 feature의 추세가 비슷한 형태를 보이며, 특히 month와 2month의 class에서는 해당 feature의 값 자체가 매우 유사한 것을 확인할 수 있음
- 즉, 비용 대비 라이트 레이드 완료 확률 feature를 이용해 month와 2month class를 구분하는 것은 불가능함

결과 및 해석 1

- 단일 feature 중요도 실험



< 시간 대비 라이트 레이드 완료 확률 >

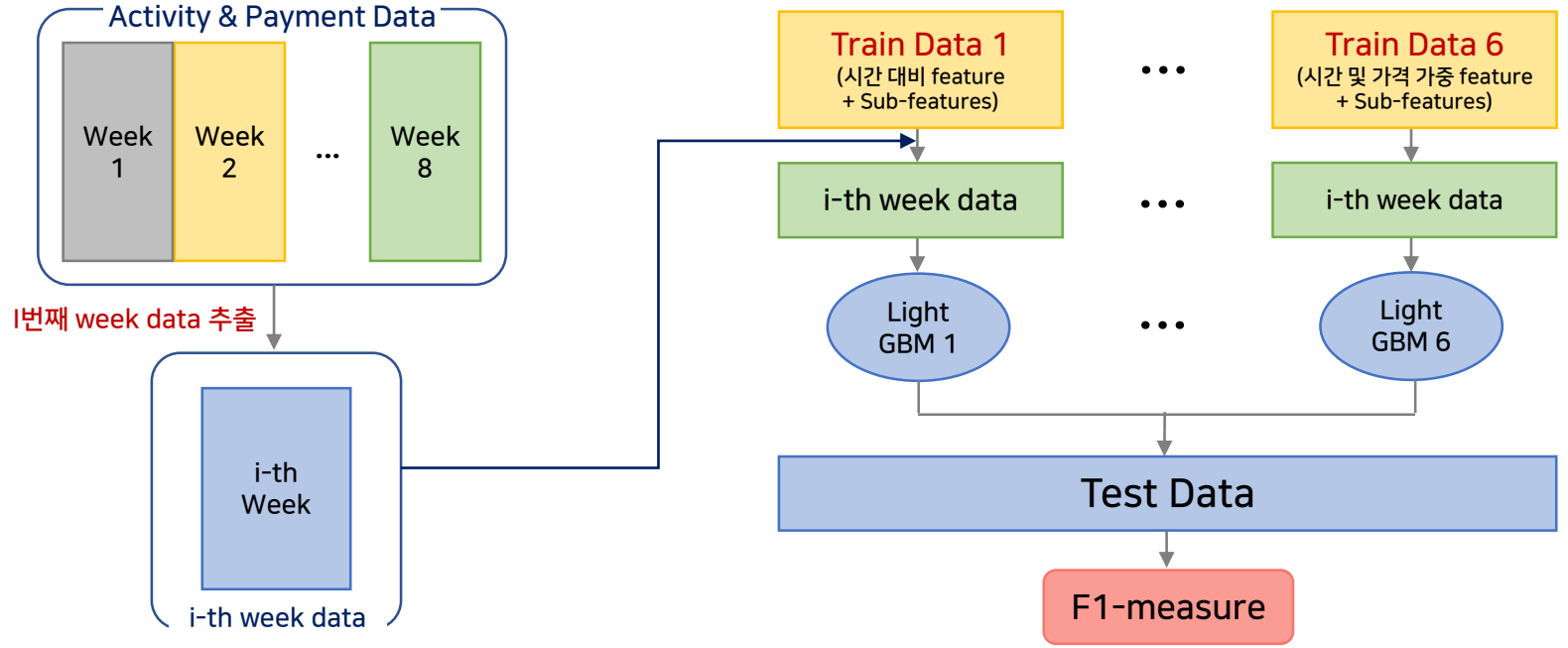


< 비용 대비 라이트 레이드 완료 확률 >

- 반면, 시간 대비 라이트 레이드 완료 확률은 4개 class에서 모두 각기 다른 형태를 보임. Month와 2month class는 비슷한 추세를 보이지만, feature 값 자체에는 차이가 존재하는 것을 확인 할 수 있음. 이러한 현상은 전투 관련 feature들에서 공통적으로 나타남
- 즉, 이탈/잔존 분류에서는 시간 및 비용 대비 feature의 중요도에 큰 차이가 없지만, 언제 이탈할지(week, month, 2month)를 결정하는 데에는 전투 관련 feature에 한해서는 비용보다 시간과 관련된 feature가 더 중요하다고 볼 수 있음

결과 및 해석 2

- 1~8 Week Data 중요도 실험



- Step 1) 각 train data에서 i번째 week data를 추출함
- Step 2) 데이터를 기반으로 6개의 모델로 구성된 앙상블 모델을 학습함
- Step 3) Step 2에서 학습한 모델을 test data에 적용해 앙상블 모델의 F1-measure를 측정함

결과 및 해석 2

- 1~8 Week Data 중요도 실험

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8
F1-measure	0.2012	0.2342	0.2433	0.2547	0.3948	0.4183	0.6381	0.6874

- 특정 week data만으로 학습한 모델을 test data에 적용해 산출한 F1-measure 값은 위와 같음
- Week 1~4의 단일 week 데이터로 학습한 모델은 성능이 매우 저조함
- Week 7~8의 단일 week 데이터로 학습한 모델은 한 주간의 데이터만으로도 준수한 결과를 보임
- 따라서, 이탈 예측 시점과 가까운 week에 수집한 데이터일수록 이탈 예측에 더 중요한 데이터라고 볼 수 있음

결과 및 해석 2

- 1~8 Week Data 중요도 실험

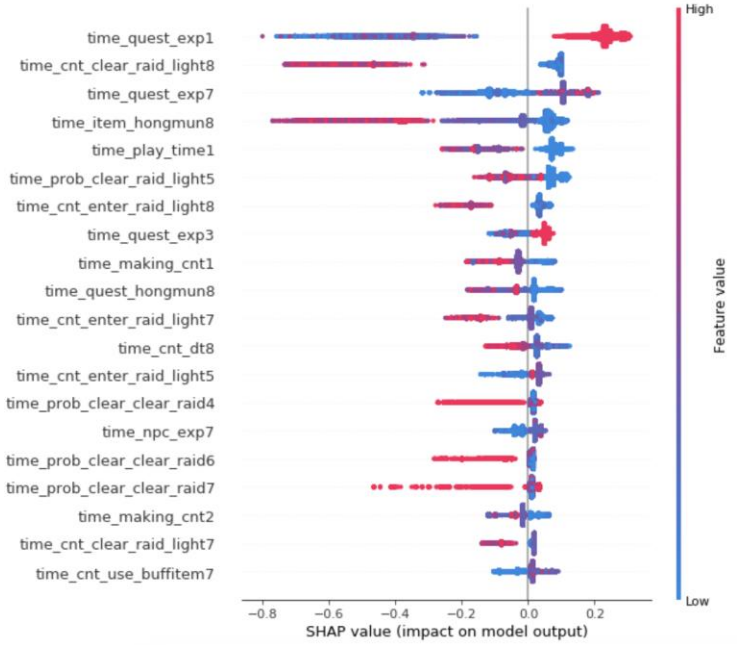
	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8
F1-measure	0.2012	0.2342	0.2433	0.2547	0.3948	0.4183	0.6381	0.6874

- 하지만, Week 7~8의 단일 week 데이터로 학습한 모델보다 week 1~8의 전체 데이터로 학습한 모델의 성능이 더 좋으므로 week 1~4의 데이터도 이탈 예측에 중요한 역할을 한다고 볼 수 있음
- 즉, week1~4의 데이터는 단일 데이터로는 좋지 못한 성능을 보이지만, week5~8 데이터와 결합함으로 인해 이탈 예측에 중요한 역할을 함

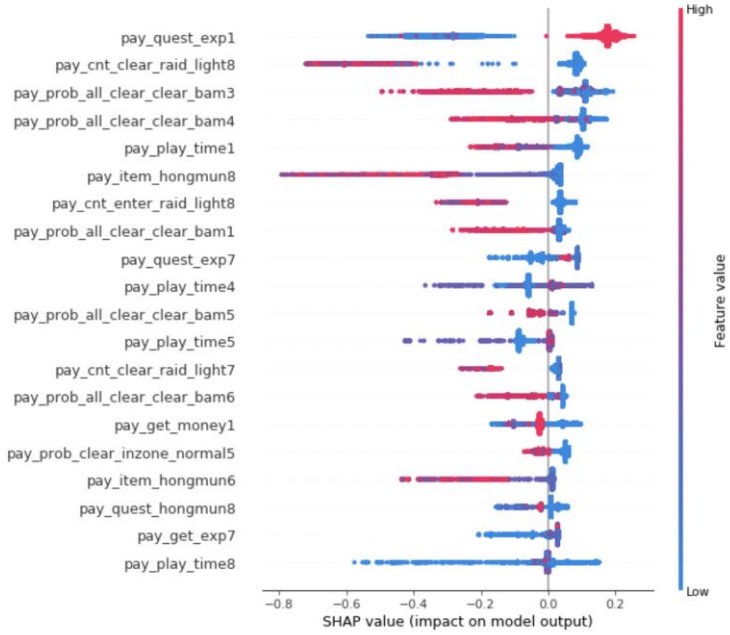
SHAP를 이용한 분석 시각화

- 1. 이탈 원인 유추를 위한 이탈/잔존 Binary모델 구축
- 2. 중요 변수 추출 및 SHAP value 시각화
- 3. 중요 변수 그룹화 및 특징 추출 (경험치, 시간, 전장 관련 feature로 그룹화)

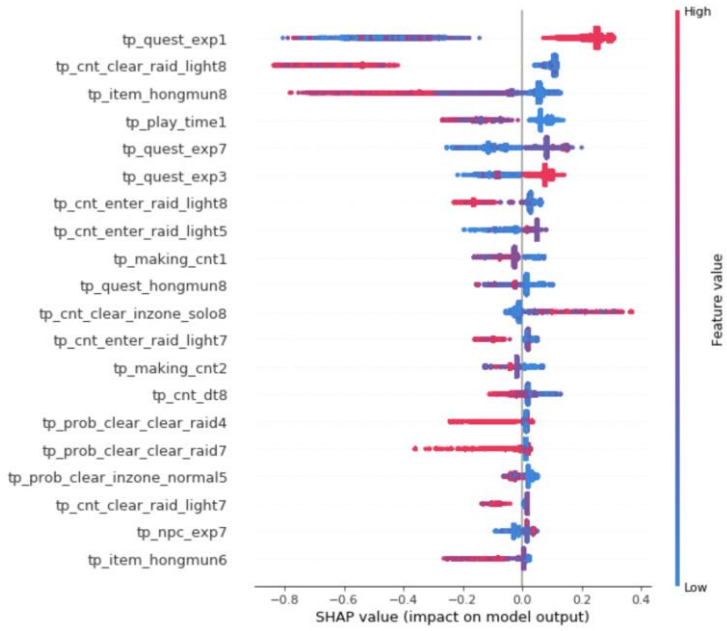
* SHAP values : 각 feature가 model의 output에 미치는 영향의 정도를 나타냄



< 시간 대비 feature SHAP value 시각화 >



< 비용 대비 feature SHAP value 시각화 >

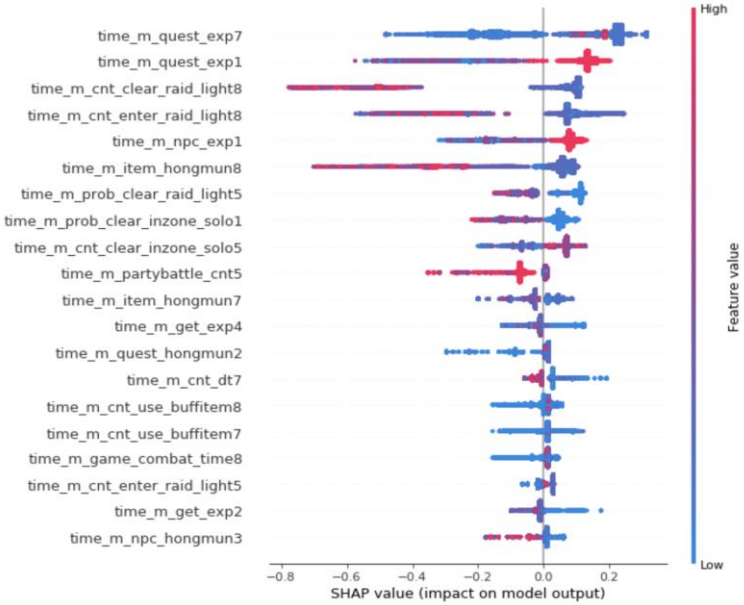


< 시간 및 비용 대비 feature SHAP value 시각화 >

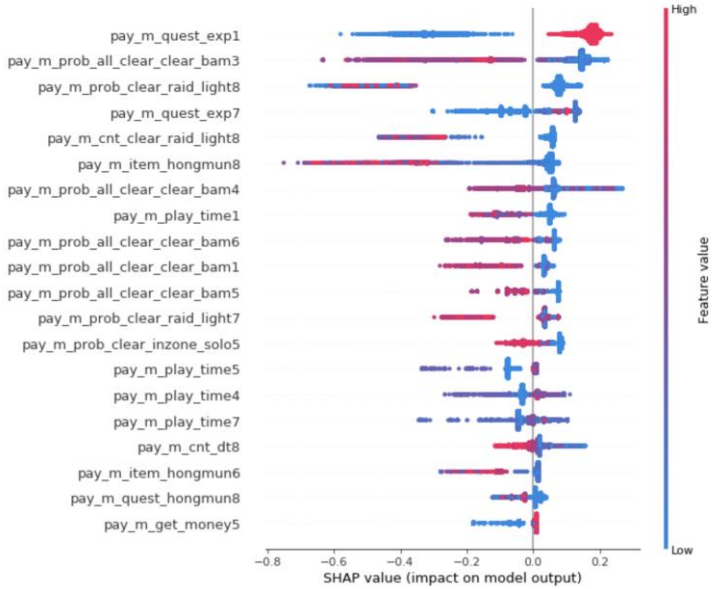
SHAP를 이용한 분석 시각화

- 1. 이탈 원인 유추를 위한 이탈/잔존 Binary모델 구축
- 2. 중요 변수 추출 및 SHAP value 시각화
- 3. 중요 변수 그룹화 및 특징 추출 (경험치, 시간, 전장 관련 feature로 그룹화)

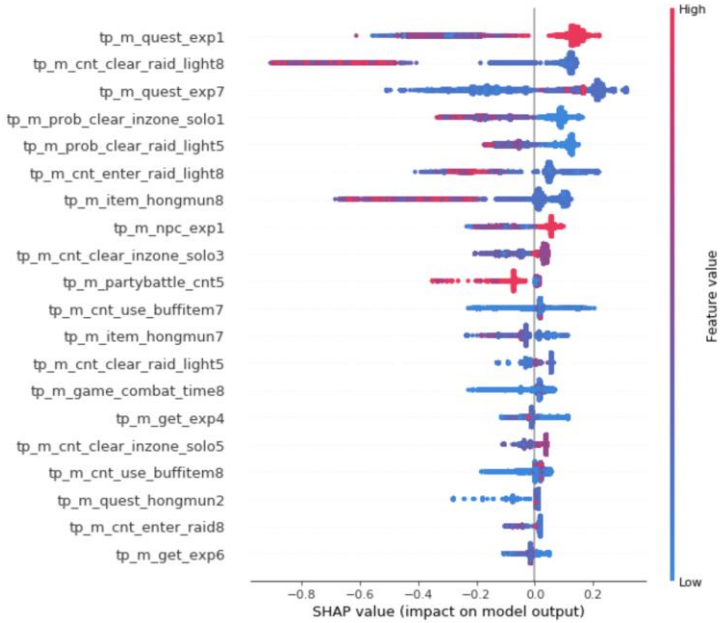
* SHAP values : 각 feature가 model의 output에 미치는 영향의 정도를 나타냄



< 시간 가중 feature SHAP value 시각화 >



< 비용 가중 feature SHAP value 시각화 >



< 시간 및 비용 가중 feature SHAP value 시각화 >

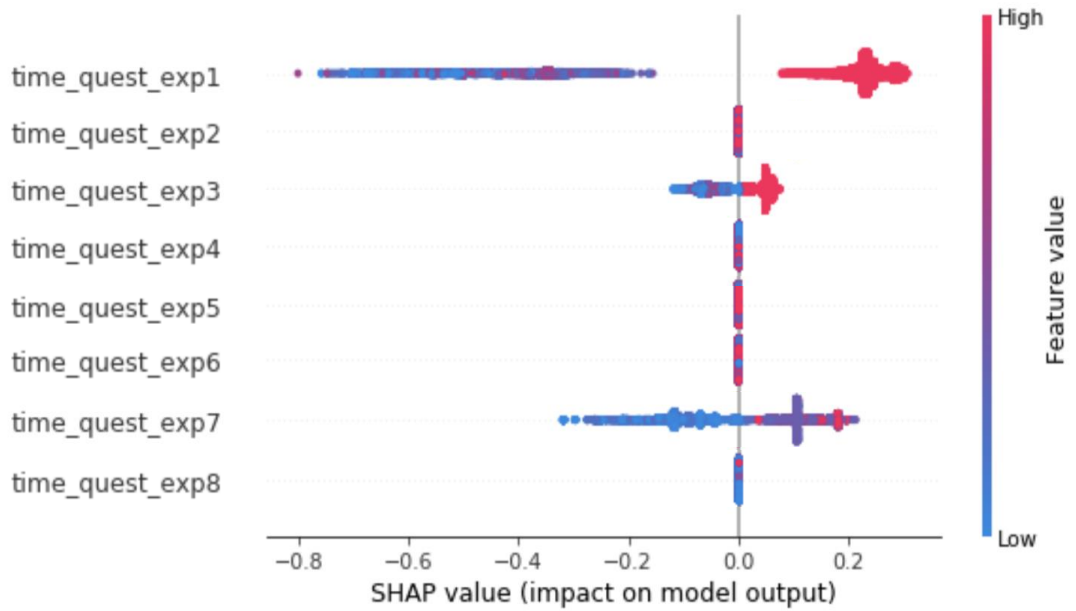
SHAP를 이용한 분석 시각화

- 중요 변수 추출 및 SHAP value 시각화를 통한 feature 특징 분석

  - 1. 이전 분석 결과와 비슷하게 7, 8주차 feature들이 주로 중요 변수로 추출됨
  - 2. 1, 2주차 feature 또한 중요 변수로 추출되는 것을 확인할 수 있음
  - 3. 단일 feature에 대한 SHAP value 해석은 유의미하지 않음
    - ex) 1주차에 시간 대비 얻는 경험치가 크면 클수록 이탈할 확률이 높음
  - 4. 그룹별로 다양한 feature들이 추출됨
    -> 게임시간, 경험치, 전장 feature로 그룹화한 후, SHAP value 시각화 및 상세 분석 필요

SHAP를 이용한 분석 시각화 – Group 1

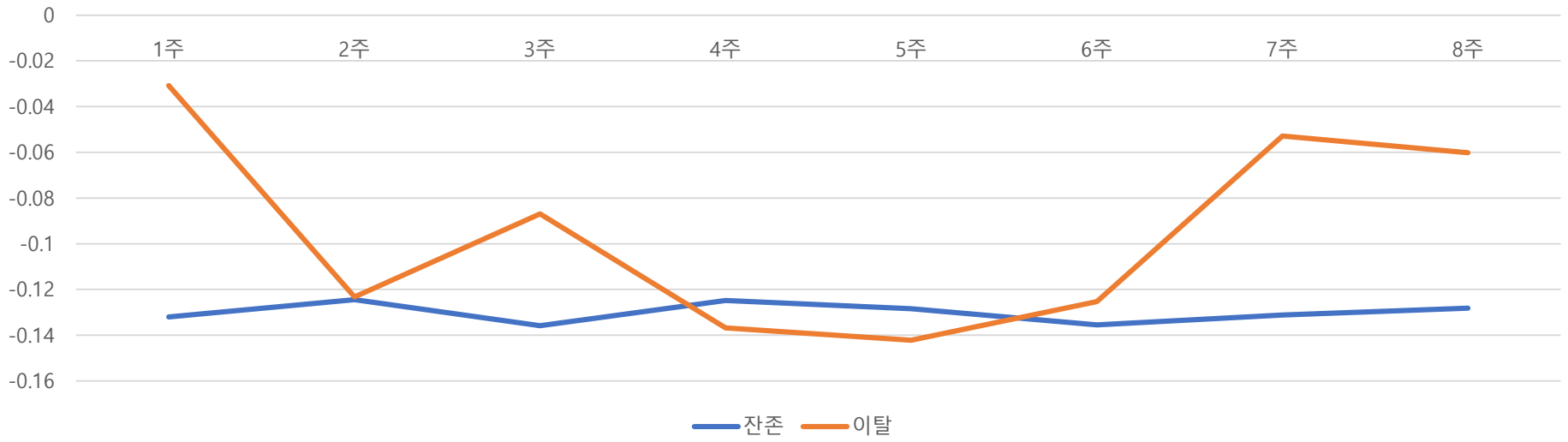
- 경험치 그룹 feature (시간 대비 quest_exp)



- 1,3,7주차의 시간 대비 퀘스트 일반 경험치가 중요한 feature로 추출됨
- SHAP value를 통해 첫째주에 경험치를 많이 얻었으나 시간이 지날수록 획득 경험치가 감소할수록 이탈할 확률이 높다고 해석할 수 있음

SHAP를 이용한 분석 시각화 – Group 1

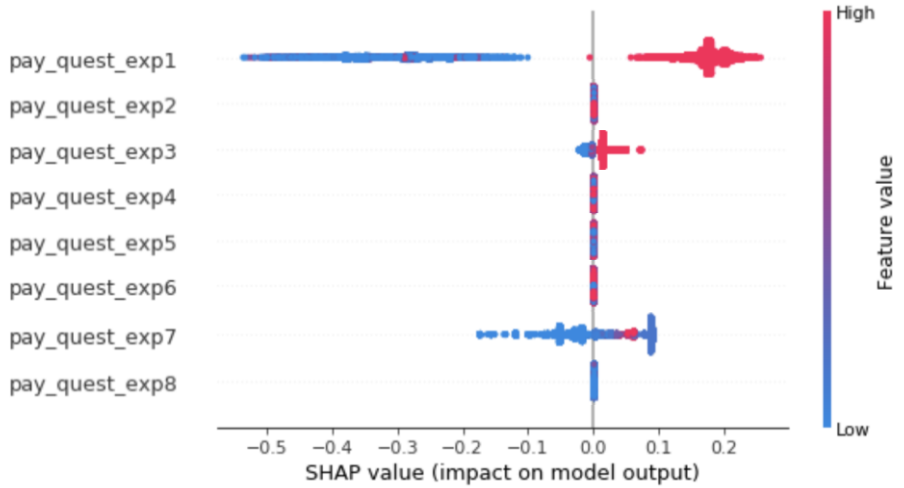
- 경험치 그룹 feature (시간 대비 quest_exp)



- 실제로 잔존/이탈 별 퀘스트 일반 경험치를 비교한 결과, 잔존의 경우 시간에 따른 획득 경험치에 큰 변화가 없는 반면, 이탈의 경우 큰 변동성을 보임
- 앞의 SHAP 분석 결과에서 8주차의 quest_exp가 중요 변수로 추출되지 않은 원인은 해당 feature가 7주차의 feature와 correlation이 높지만 7주차의 값이 더 크기 때문으로 추정됨

SHAP를 이용한 분석 시각화 – Group 1

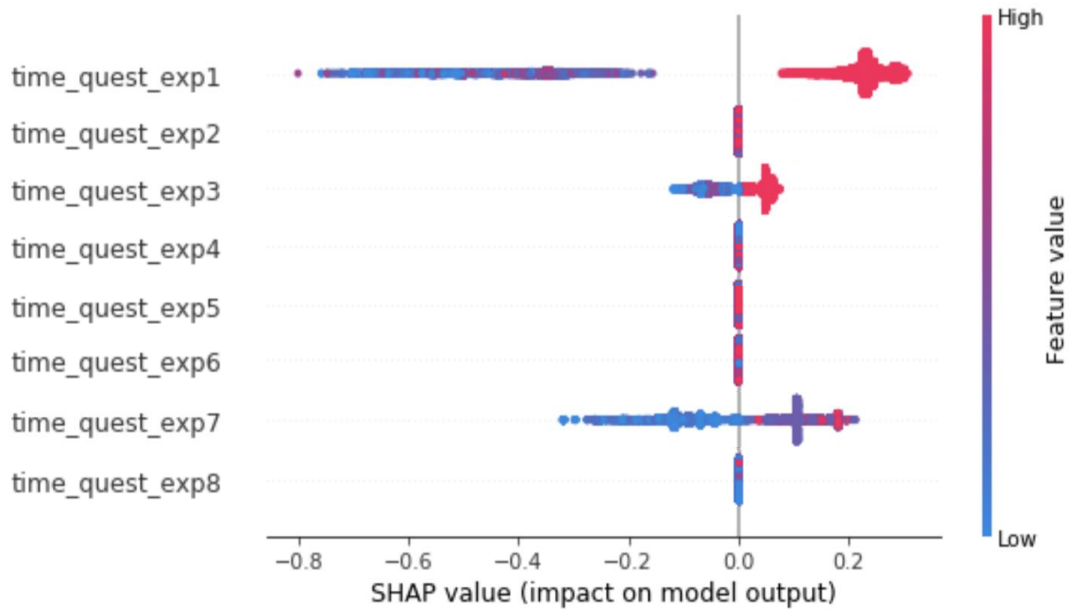
- 경험치 그룹 feature (비용 대비 quest_exp)



- 시간 대비 quest_exp 분석 결과와 유사한 결과를 보임
- 이러한 현상은 시간 및 지불의 대비 및 가중 feature에서 공통적으로 나타남
- 즉, 시간에 따른 획득 경험치 변화가 적은 유저는 이탈할 확률이 적으며, 변동성이 큰 유저는 이탈할 확률이 높고 특히 경험치가 7주차까지 증가하다 8주차에 감소하는 유저는 이탈할 확률이 더 높음

SHAP를 이용한 분석 시각화 – Group 2

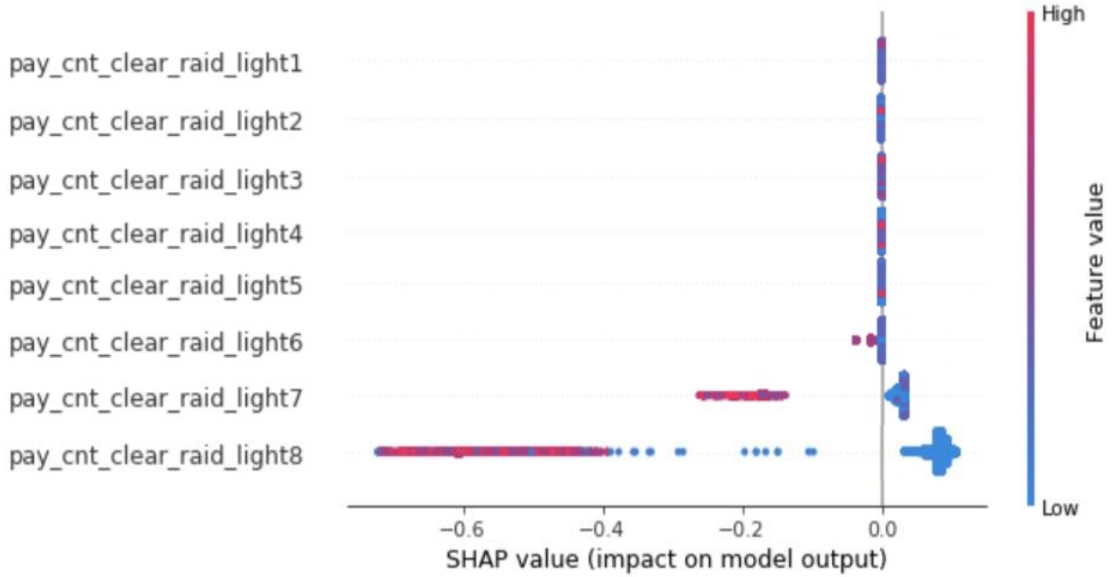
- 게임시간 그룹 feature (시간 대비 cnt_dt)



- 게임에 투자한 시간이 주차에 따라 변동성이 크고 최근 2주 내에 감소하는 유저는 이탈할 확률이 높음
- 투자한 시간은 feature 그룹 별로 같으므로 모든 그룹에서 비슷한 결과가 도출됨

SHAP를 이용한 분석 시각화 – Group 3

- 전장 그룹 feature (비용 대비 cnt_clear_raid_light)



- 투자한 비용 대비 전장에서의 승률이 낮아질수록 이탈할 확률이 높아지나, 7-8주차의 feature만 영향을 미치는 것으로 보임
- 앞의 결과와 비교하면, 시간 대비 전장에서의 승률이 비용 대비 전장에서의 승률보다 더 많은 영향을 미치는 것을 확인할 수 있음. 이는 앞서 단일 feature별 모델 결과와 동일한 결과임

최종 결론

- 분석 결론
  - 1-4주차의 feature는 단일 사용으로는 모델의 성능이 저조하나, 7-8주차와의 feature들과 결합해 사용했을 때 중요한 역할을 함. 즉, 특정 기간의 feature의 특성이 이탈에 영향을 끼치는 것이 아닌 시간에 따른 feature의 변화가 이탈에 영향을 미치는 주요한 요인임
  - 전장 관련 feature에 대해서는 비용보다 투자한 시간과 관련된 feature가 이탈에 영향을 미침
  - 투자한 시간대비 전장의 승률이 감소하는 것은 레벨이 증가하면서 전장 및 던전의 난이도가 높아지기 때문에 자연스러운 현상으로 볼 수 있음. 시간에 따라 투자한 시간 대비 전장의 승률이 지속적으로 감소할 경우, 빠른 시일 내에 이탈(week)할 확률이 높음

최종 결론

- 분석 결론
  - 시간이 지남에 따라 얻는 경험치가 일정할 경우, 잔존할 확률이 높으며 변동성이 큰 경우 이탈할 확률이 높음. 투자 대비 얻는 경험치가 증가하다 감소할 경우, 게임에 대한 흥미도가 감소하는 것으로 해석됨
  - 게임에 투자한 시간 자체가 감소할 경우 이탈 확률이 높아짐
  - 모든 개별 feature에 대해 초반(1,2주차)와 후반(7,8주차)의 feature가 중요 변수로 추출되는 현상이 공통적으로 나타남. 즉, 중간(3~6주) feature보다 초반과 후반의 feature 변화에 따라 이탈 여부가 결정됨

최종 결론

- 유저의 이탈 방지 Idea 제안
  - 유저의 게임 시작 이후 7주 이후부터 집중 모니터링이 필요함
  - 비용과 별개로 투자한 시간 대비 얻을 수 있는 보상의 증가 필요함
  - 투자한 시간 대비 전장 및 던전 승률이 감소하는 유저의 경우 이탈 고위험군 분류 및 조치가 필요함
  - 게임에 투자한 시간 자체가 감소하는 유저의 경우 이탈 위험군 분류 및 조치가 필요함

감사합니다